Equivalence Checking in Embedded Systems Design Verification

Soumyadip Bandyopadhyay

soumyadip@cse.iitkgp.ernet.in

1 Introduction

In this paper we focus on some aspects related to modeling and formal verification of embedded systems. Many models have been proposed to represent embedded systems [1] [2]. These models encompass a broad range of styles, characteristics, and application domains and include the extensions of finite state machines, data flow graphs, communication processes and Petri nets. In this report, we have used a PRES$^+$ model (Petri net based Representation for Embedded Systems) as an extension of classical Petri net model that captures concurrency, timing behaviour of embedded systems; it allows systems to be representative in different levels of abstraction and improves expressiveness by allowing the token to carry information [3]. This modeling formalism has a well defined semantics so that it supports a precise representation of system. As a first step, we have taken an untimed PRES$^+$ model which captures all the features of PRES$^+$ model except the time behaviour which have reported in earlier report.

A typical synthesis flow of complex systems like VLSI circuits or embedded systems comprises several phases. Each phase transforms/refines the input behavioural specification (of the systems to be designed) with a view to optimize time and physical resources. Behavioural verification involves demonstrating the equivalence between the input behaviour and the final design which is the output of the last phase. In computational terms, it is required to show that all the computations represented by the input behavioural description, and exactly those, are captured by the output description.

Modeling using PRES$^+$, as discussed above, may be convenient for specifying the input behaviour because it supports concurrency. However, there is no equivalence checking method reported in the literature for PRES$^+$ models to the best of our knowledge. In contrast, equivalence checking of FSMD models exist [4]. Although Transformation procedure from non-pipelined version PRES$^+$ to pipelined version PRES$^+$ is reported [3]. As a first step, we seek to hand execute our reported algorithm on a real life example and we have to translate two versions of PRES$^+$ models to FSMD models.

The rest of the paper is organized as follows. Section 2 presents the definition of PRES$^+$ and FSMD models. Section 3 presents Proposed algorithm for conversion from an untimed PRES$^+$ models to an FSMD models. Section 4 presents notion of equivalence, abstraction. In this section we have also presented the working principal of an example of real life embedded systems. Section 5 verify the equivalence between initial and transformed behaviour using FSMD equivalence checking method. Finally, some future works are identified in Section 6

2 Brief description of PRES$^+$ and FSMD model

Before the conversion mechanism we discuss the design representation of PRES$^+$ models.

2.1 Description of PRES$^+$ models

A PRES$^+$ model is a seven tuple $N = (P, V_P, K, T, I_P, O, M_0)$, where the members are defined as follows. The set $P = \{p_1, p_2, ..., p_m\}$ is a finite non-empty set of places; V_P: the set of variables. A place p is associated with

a variable v_p; therefore, $V_P = \{v_p \mid p \in P\}$. Every place is capable of holding a token having a value. A token value may be of any type, such as, Boolean, integer, etc., or a user-defined type of any complexity (for instance, a structure, a set, or a record). The set K denotes the set of all possible token types. Thus, K is a set of sets. The set $T = \{t_1, t_2, ..., t_n\}$ is a finite non-empty set of transitions; $I_P \subseteq P \times T$ is a finite non-empty set of input arcs which define the flow relation from places to transitions $-$ "input" with respect to transitions; $O \subseteq T \times P$ is a finite non empty set of output arcs which define the flow relation from transitions to places. A marking M is the assignment of tokens to places of the net; hence, $M \subseteq P$. The marking of a place $p \in P$, denoted $M(p)$, is either 0 or 1. For a particular marking M, a place p is said to be marked iff $M(p) = 1$. M_0 is the initial marking of the net, depicting the places having tokens initially.

The type function τ: $P \rightarrow K$ associates every place $p \in P$ with a token type.

The pre-set $^\circ t$ of a transition $t \in T$ is the set of input places of t. Thus, $^\circ t = \{p \in P \mid (p,t) \in I_P\}$. Similarly, the post-set t° of a transition $t \in T$ is the set of output places of t. So, $t^\circ = \{p \in P \mid (t,p) \in O\}$ and $\forall t \in T, \forall p_1, p_2 \in t^\circ, \tau(p_1) = \tau(p_2)$ and $v_{p_1} = v_{p_2}$. The subset $V_{^\circ t} = \{v_p \mid p \in {^\circ t}\}$ is the set of variables associated with places from which input arcs lead to the transition t. Similarly, the pre-set $^\circ p$ and the post-set p° of a place $p \in P$ are given by $^\circ p = \{t \in T \mid (t,p) \in O\}$ and $p^\circ = \{t \in T \mid (p,t) \in I_p\}$, respectively.

For every transition $t \in T$, there exists a transition function f_t associated with t; that is, for all $t \in T$, f_t: $\tau(p_1) \times \tau(p_2) \times .... \times \tau(p_a) \rightarrow \tau(q)$, where $^\circ t = \{p_1, p_2, ..., p_a\}$ and $q \in t^\circ$. The functions f_t's are used to capture the functional transforms that take place of the variable associated with the output places of the transitions i.e, $v_q \Leftarrow f_t(v_{p_1}, v_{p_2}, ...v_{p_a})$.

A transition $t \in T$ may have a guard g_t associated with it. The guard of a transition t is a predicate g_t: $\tau(p_1) \times \tau(p_2) \times .... \times \tau(p_a) \rightarrow \{0, 1\}$, where $^\circ t = \{p_1, p_2, ..., p_a\}$ over the variable set $V_{^\circ t}$.

2.2 Description of FSMD model

A finite state machine with data path (FSMD) is a universal specification model. An FSMD is defined as an ordered tuple $F = (Q, q_0, I_F, V_F, O, f, h)$ where

$Q = \{q_0, q_1, ..., q_n\}$ is a finite set of control states. $q_0 \in Q$ is the reset state. I_F is the set of primary input signals. V_F is the set of storage variables. O_F is the set of primary output signals, $O_F \subseteq V_F$. f: $Q \times 2^S \rightarrow Q$ is the state transition function. h: $Q \times 2^S \rightarrow U$ is the update function of the output and the storage variables, where S and U are as defined below $S = \{L \cup E_R \mid L$ is the set of boolean literals of the form b or b, $b \in B \subseteq V$ is a boolean variable and $E_R = \{eR0 \mid e \in E_A\}\}$; its represent the set of status expression over $I_F \cup V$, where E_A represents a set of arithmetic expression over $I_F \cup U$ of input and storage variables and R is any arithmetic relation. $R \in \{=, \neq, >, \geq, <, \leq\}$. $U = \{x \Leftarrow e \mid x \in O_F \cup V_F$ and $e \in E_A \cup E_R\}$ represent set of storage or output assignment.

3 Proposed algorithm for conversion from an untimed PRES$^+$ models to an FSMD models

Let the input PRES$^+$ model be N and the generated FSMD model be F. For simplicity, we assume that all tokens are of integer type. i.e $\tau(p)$ = Z for all $p \in P$.

The first step of our algorithm computes the following entities in the FSMD model: q_0, I_F, V_F, O_F, U and S. The algorithm then goes on to compute Q: the set of states; f: the state transition function and h: the update function. Symbolic simulation of the PRES$^+$ model is used to compute these entities starting from the initial marking $M_0 = q_0$.

- At each step of the simulation, starting from a present marking $M(= q) \subseteq P$ the algorithm enumerates all the possible sets of transitions of N from M; for each of these sets of possible transitions, it constructs the next state (q^+) of F from the new marking M^+ of the PRES$^+$ model N.

- Obtain the transition from q to q^+ in F .

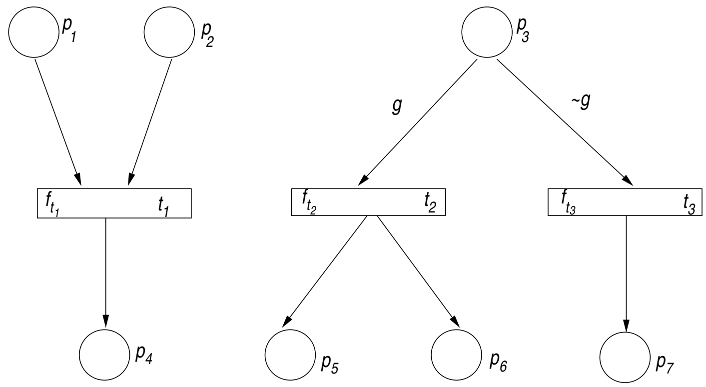

Figure 1: Places and transitions in a PRES+ model

- For example, consider the scenario given in Figure 1. Let $M = \{p_1, p_2, p_3\} = q$; so the set T_q of all transitions emanating from the places in M is given by $T_q = \{t_1, t_2, t_3\}$. The possible sets of transitions are $\{t_1, t_2\}$ leading to the marking $M_1^+ = \{p_4, p_5, p_6\} = q_1^+$ and $\{t_1, t_3\}$ leading to the marking $M_2^+ = \{p_4, p_7\} = q_2^+$. The FSMD transition $(q \rightarrow q_1^+)$ is associated with the guard condition g and the FSMD transition $(q \rightarrow q_2^+)$ is associated with the guard condition $\neg g$, i.e, $f(q, g) = q_1^+$ and $f(q, \neg g) = q_2^+$. $h(q, g) : v_{p_4} \Leftarrow f_{t_1}(v_{p_1}, v_{p_2})$ and $v_{p_6} = v_{p_5} \Leftarrow f_{t_2}(v_{p_3})$. $h(q, \neg g) : v_{p_4} \Leftarrow f_{t_1}(v_{p_1}, v_{p_2})$ and $v_{p_7} \Leftarrow f_{t_3}(v_{p_7})$.

Algorithm

Steps:

Step 1: Given PRES+ model

$q_0 \Leftarrow M_0$;

$I_F \Leftarrow \{$ Variables associated with $p \mid p \in M_0(p)\}$;

$V_F \Leftarrow \{$Variables associated with $p \mid p \notin M_0(p)\}$;

// O_F is the set of variables associated with places from which no arcs are input // to any transition. Therefore

$O_F \Leftarrow \{$Variable associated with $p \mid p^\circ = \phi\}$;

// U is obtain from transition function of PRES+ model and variable associated // with post set of that transition. Therefore,

$U \Leftarrow \{x \Leftarrow f_t^n(v_1, v_2, ..., v_n) \mid t \in T, f_t^n$ is the function associated with t, $x = v_{t^\circ}$ and $v_i \in v_{\circ t}, 1 \leq i \leq n\}$;

// S is obtained from the guard conditions of the PRES+ models. Therefore,

$S \Leftarrow \{g_t \mid t \in T\}$;

Step 2: $Q \Leftarrow \{q_0\}$; $Q_{new} \Leftarrow Q$; $Q^+_{new} \Leftarrow \emptyset$;

Step 3: $\forall q \in Q_{new}$

Step 3.1: $Q_{new} \Leftarrow Q_{new} - \{q\}$; $T_q \Leftarrow \{t \mid {}^\circ t \in q\}$;
$\tau_q \Leftarrow$ constructSetOfTransitions (T_q); // $\tau_q \in 2^{T_q}$, the set of possible
// transitions.
$Q^q_{new} = \emptyset$, empty set, //Q^q_{new}: the set of next states generated
// depending on q mutually exclusive
// depending on guard condition
// associated with member of τ_q.

Step 3.2: $\forall T \in \tau_q$

Step 3.2.1: $q^+_T \Leftarrow \{t \mid t_i \in T\}$; $Q_{new} \Leftarrow Q^q_{new} \cup \{q^+_T\}$;

Step 3.2.2: Let G_T be the set of guards associated with $t \in T$. In the table of the function f, insert entry
$f(q, G_T) = q^+$

Step 3.2.3: Let A_T be the set of assignments of the form
$\{v \Leftarrow f_t(v_1, v_2, ..., v_n) \mid t \in T, \{v\} = t^\circ, \{v_1, v_2, ..., v_n\} =^\circ t$
and f_t is the function associated with t $\}$;
In the table of the function h, insert the entry $h(q, G_t) = A_T$;
// members of A_T are carried out in parallel

Step 3.2.4: $Q^+_{new} \Leftarrow Q^+_{new} \cup Q^q_{new}$;

Step 4: // Any new state generated
$Q^+_{new} \Leftarrow Q^+_{new} - Q$;
if $Q^+_{new} = \emptyset$ exit;
else $\{$ $Q \Leftarrow Q \cup Q^+_{new}$; $Q_{new} \Leftarrow Q^+_{new}$; $Q^+_{new} \Leftarrow \emptyset$;
goto *Step 3*
$\}$

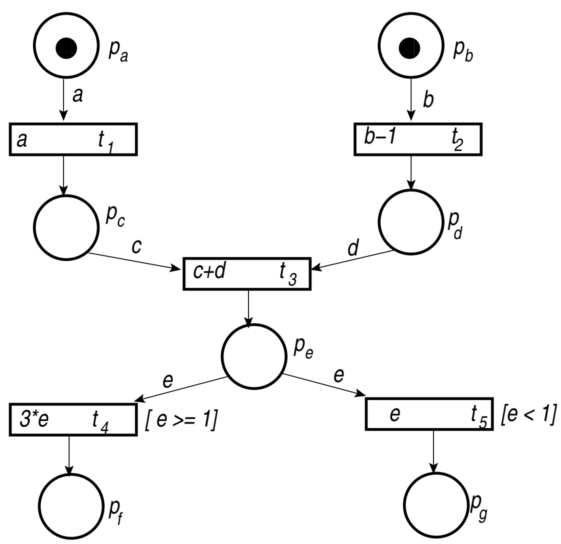

Figure 2: PRES+ model to be converted into FSMD model

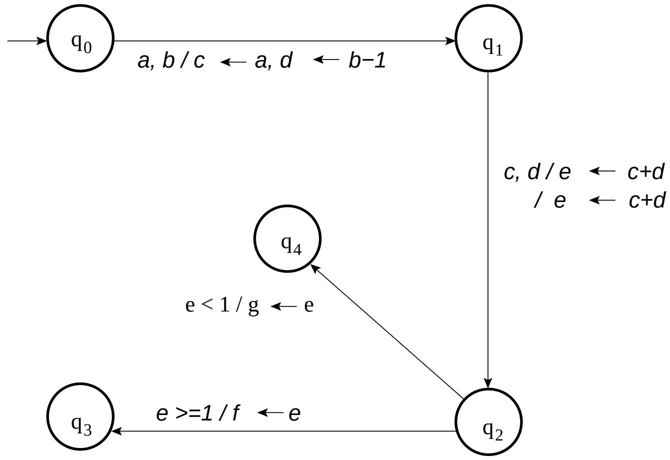

Figure 3: FSMD model equivalent to the PRES+ model of Figure 1

4 Notion of equivalence and Real life example

4.1 Notion of equivalence between two PRES+ models

In the synthesis process there are a number of refinement phase. System model is transformed in each phases. So the validity of this transformation depends on the equivalence between the input behaviour and the output behaviour of each phase. Literature [3] has propounded three notion of equivalence - cardinality equivalence, functional equivalence, and time equivalence; the two PRES+ models are totally equivalence iff they satisfies all these equivalence. We are dealing with untimed PRES+ hence, there is no need to show time equivalence. Two PRES+ models N_1 and N_2 are cardinality equivalence iff:

1. There exist a one to one correspondence between the in-ports and the out-ports of N_1 and N_2 i.e f_{in}: $inP_1 \leftrightarrow inP_2$ and f_{out}: $outP_1 \leftrightarrow outP_2$.
2. The Initial markings $M_{1,0}$ and $M_{2,0}$ of N_1 and N_2 are the same.
3. After execution of N_1 and N_2 if the tokens are accumulated at out-ports of the each nets, there is a one to one correspondence of marking at their out-ports.

For example in Figure 4 inP_1 = $\{P_a, P_b\}$, outP_1 = $\{P_e, P_f, P_g\}$, inP_2 = $\{P_{aa}, P_{bb}\}$ outP_2 = $\{P_{ee}, P_{ff}, P_{gg}\}$ and f_{in} and f_{out} are defined by $f_{in}(P_a) = P_{aa}$, $f_{in}(P_b) = P_{bb}$, $f_{out}(P_e) = P_{ee}$, $f_{in}(P_f) = P_{ff}$ and $f_{in}(P_g) = P_{gg}$. Second condition also satisfies the two nets. N_1 and N_2 also satisfies third condition i.e after execution of N_1 and N_2 all out-ports of N_1 and N_2 contains token and they are one to one correspondence. Hence two PRES+ N_1 and N_2 are cardinality equivalence.

Two nets PRES+ N_1 and N_2 are functionally equivalent iff:

1. N_1 and N_2 are cardinality equivalent,
2. The token values in out-ports in N_1 and N_2 are the same.

For example in Figure 5 N_1 and N_2 are cardinality equivalence. If P_a of N_1 and P_{aa} of N_2 contain token whose values are 2. then after execution of N_1 and N_2 the out-port of N_1 and N_2 contains token whose values are 5. Hence two nets are totally equivalence.

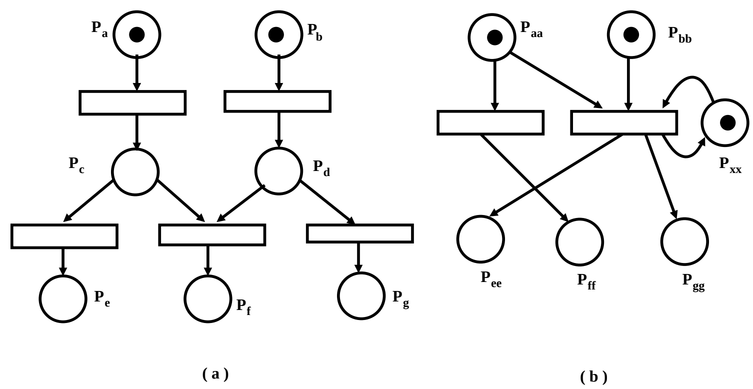

Figure 4: Cardinality equivalence nets

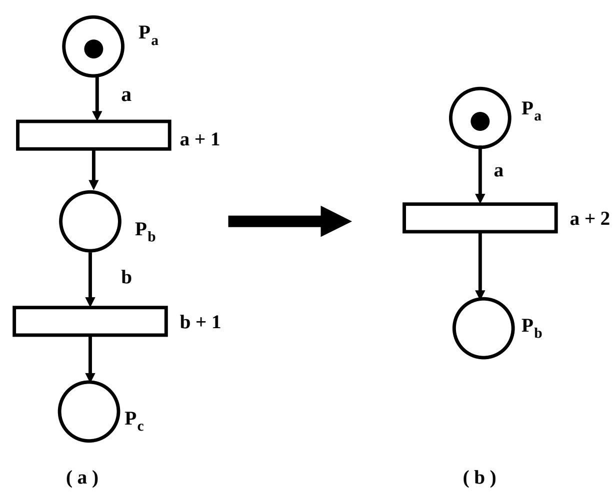

Figure 5: Functional equivalence nets

4.2 Modeling of a real life example

Non-pipelined pipelined version of PRES+ model for a jammer is reported [3]. Transformation technique from non-pipelined version of PRES+ model to pipeline version of PRES+ model also have been reported [3]. Non-pipelined and pipelined version of PRES+ models are shown in Figure 6 and Figure 7 respectively.

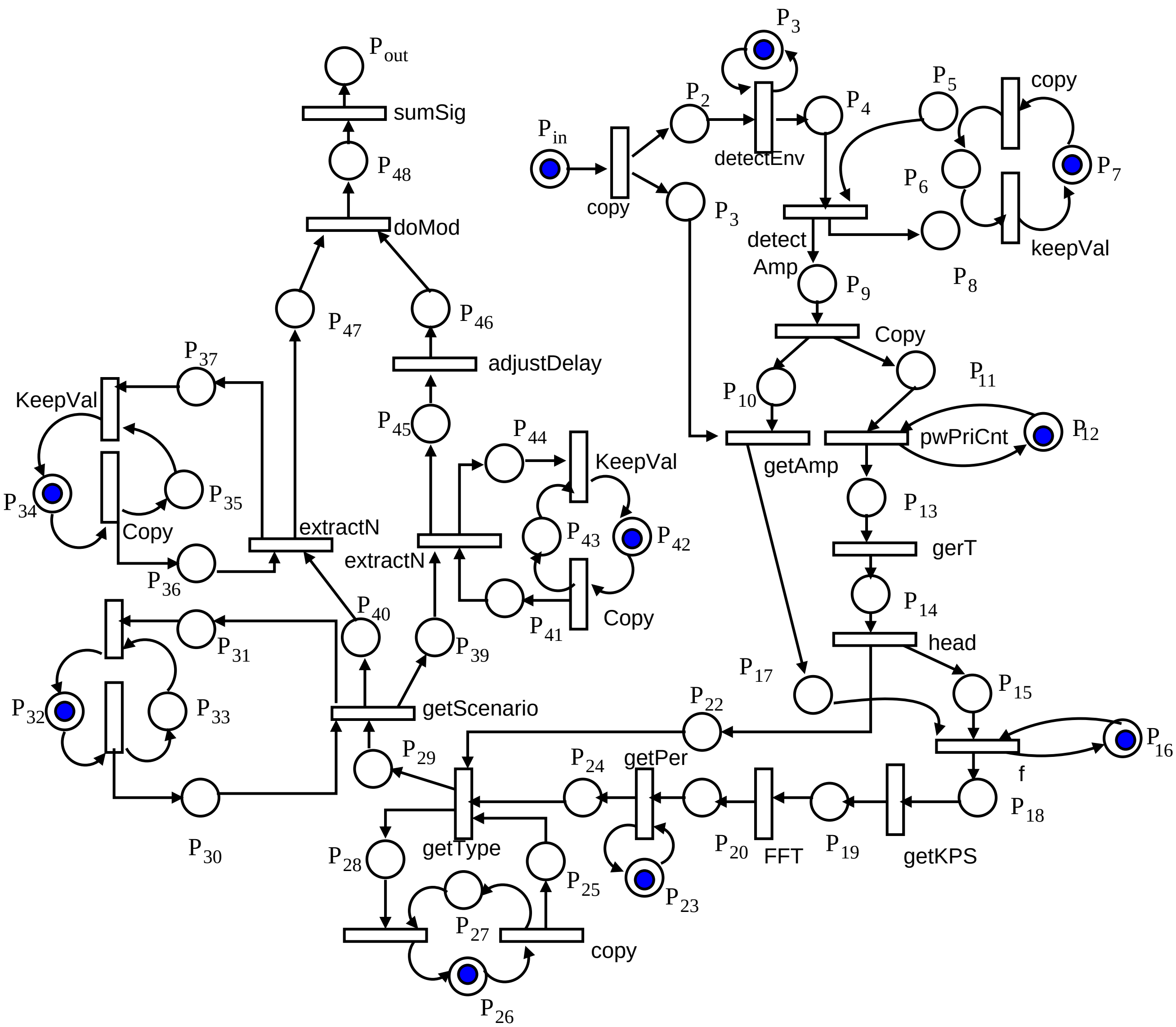

Figure 6: A non pipelined PRES+ model for a jammer

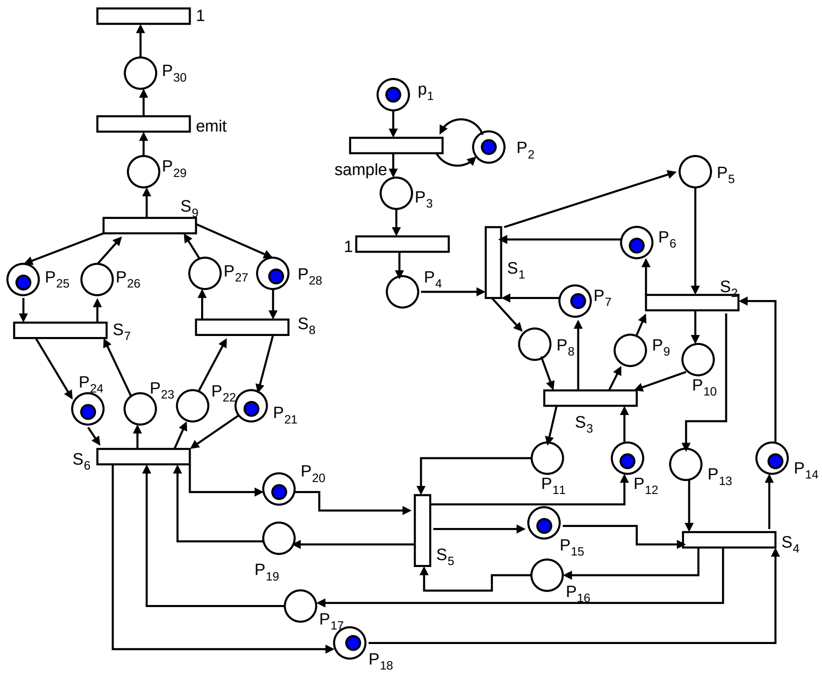

Figure 7: A pipelined PRES+ model for a jammer

5 Experimental results

We have reported a translation algorithm from untimed PRES+ model to FSMD model. Hand execution of this translation algorithm we have get FSMD model of the jammer from non pipelined PRES+ model. The FSMD model is given Figure 8 and transition function is given in Table 1. Similarly, the FSMD generated from the pipelined PRES+

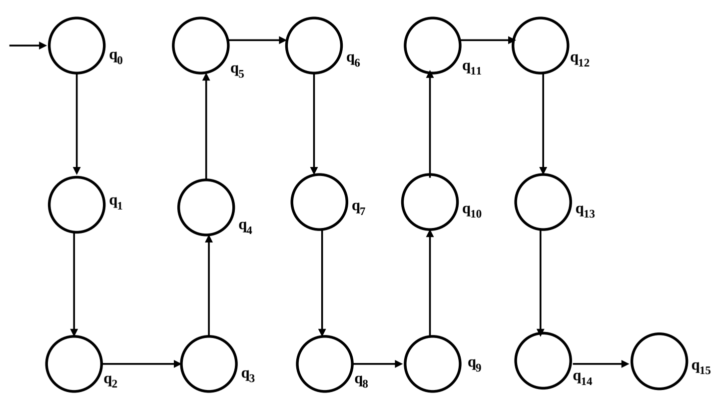

Figure 8: A non pipelined FSMD model for a jammer

model is shown in Figure 9 and the state transition function given in Table 2

State	*Transition function*
$\langle\ q_0, q_1\ \rangle$	in-Copy, Thresold-copy, trigerselect-Copy, opMode-Copy, modParLib-Copy and delayPerLib-copy
$\langle\ q_1, q_2\ \rangle$	detectEnv
$\langle\ q_2, q_3\ \rangle$	detectAmp
$\langle\ q_3, q_4\ \rangle$	thresold-keepVal, copy
$\langle\ q_4, q_5\ \rangle$	getAmp, pwPricnt
$\langle\ q_5, q_6\ \rangle$	getT
$\langle\ q_6, q_7\ \rangle$	head
$\langle\ q_7, q_8\ \rangle$	f
$\langle\ q_8, q_9\ \rangle$	getKPS
$\langle\ q_8, q_9\ \rangle$	FFT
$\langle\ q_8, q_9\ \rangle$	getPer
$\langle\ q_9, q_{10}\ \rangle$	getType
$\langle\ q_{10}, q_{11}\ \rangle$	trigSelect-keepVal, getScenario
$\langle\ q_{11}, q_{12}\ \rangle$	trigSelect-copy, opMode-keepVal, extractN, extractN
$\langle\ q_{12}, q_{13}\ \rangle$	opmode-copy, delayPerLib-keepVal, modPerLib-keepVal, adjustdelay
$\langle\ q_{13}, q_{14}\ \rangle$	delayPerLib-copy, modPerLib-copy, doMod
$\langle\ q_{14}, q_{15}\ \rangle$	sumsig

Table 1: Transition function for FSMD model obtain from normal PRES+ model of a jammer

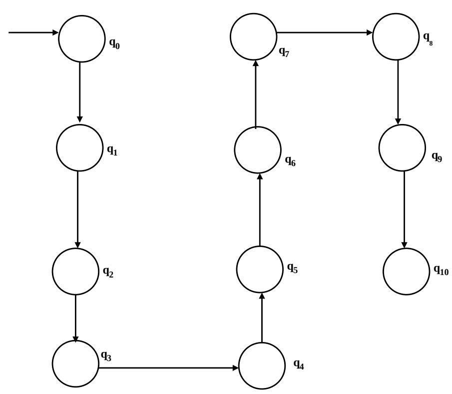

Figure 9: A pipelined FSMD model for a jammer

State	*Transition function*
$\langle\ q_0, q_1\ \rangle$	in-Copy ◇ detectEnv
$\langle\ q_1, q_2\ \rangle$	Thresold-copy ◇ keepVal ◇ detectAmp
$\langle\ q_2, q_3\ \rangle$	in-Copy ◇ getAmp
$\langle\ q_3, q_4\ \rangle$	pwPriCnt ◇ getT ◇ head
$\langle\ q_4, q_5\ \rangle$	f ◇ getKPS ◇ FFT ◇ getPer
$\langle\ q_5, q_6\ \rangle$	trigerselect-Copy ◇ keepVal ◇ getType ◇ opMode-Copy ◇ keepVal ◇ getScenario
$\langle\ q_6, q_7\ \rangle$	modParLib-Copy ◇ keepVal ◇ extractN and delayParLibCopy ◇ keepVal◇ extranctN ◇ adjustDelay
$\langle\ q_7, q_8\ \rangle$	doMod ◇ sumsig
$\langle\ q_8, q_9\ \rangle$	emit

Table 2: Transition function for FSMD model obtain from pipelined PRES+ model of a jammer

Here the FSMD equivalence checking is very straightforward. Two versions of FSMDs have only one path and the data transformation which have been shown in Table 1 and Table 2 are same. Hence two FSMD models are equivalent.

6 Plan of Future work

Carrying out analysis for correctness of technique, complexity analysis, etc. Direct equivalence checking between two PRES$^+$ models Generalization of FSMD models to timed FSMD models. We will generalize an FSMD model to timed FSMD model which can capture data path as well as timing behaviour and Conversion of PRES$^+$ models to timed FSMD models.

References

[1] S. Edwards, L. Lavagno, E. A. Lee, and A. Sangiovanni-Vincentelli, "Design of embedded systems: Formal models, validation, and synthesis," in *Proceedings of the IEEE*, pp. 366–390, 1997.

[2] P. Eles, K. Kuchcinski, Z. Peng, A. Doboli, and P. Pop, "Scheduling of conditional process graphs for the synthesis of embedded systems," in *DATE '98: Proceedings of the conference on Design, automation and test in Europe*, (Washington, DC, USA), pp. 132–139, IEEE Computer Society, 1998.

[3] L. A. Cortés, P. Eles, and Z. Peng, "Verification of embedded systems using a petri net based representation," in *ISSS '00: Proceedings of the 13th international symposium on System synthesis*, (Washington, DC, USA), pp. 149–155, IEEE Computer Society, 2000.

[4] C. Karfa, D. Sarkar, C. Mandal, and P. Kumar, "An equivalence-checking method for scheduling verification in high-level synthesis," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 27, no. 3, pp. 556–569, 2008.